Ollscoil Mhá Nuad
Maynooth University

SEMESTER 2
2022-2023

CS603
Rigorous Software Development

Prof. O. Conlan, Dr. J. Timoney, Prof. R. Monahan

Time allowed: 3 hours

Answer at least ***three*** questions

Your mark will be based on your best ***three*** answers

All questions carry equal marks

Instructions

	Yes	No	N/A
Formulae and Tables book allowed *(i.e. available on request)*		X	
Formulae and Tables book required *(i.e. distributed prior to exam commencing)*		X	
Statistics Tables and Formulae allowed *(i.e. available on request)*		X	
Statistics Tables and Formulae required *(i.e. distributed prior to exam commencing)*		X	
Dictionary allowed *(supplied by the student)*		X	
Non-programmable calculator allowed		X	
Students required to write in and return the exam question paper		X	

[25 marks]

1 (a) Explain how *Design by Contract* guides the verification of: [8 marks]

i. class methods
ii. object constructors, and
iii. overridden methods in a class inheritance.

Your explanation should include a coded example of each.

(b) Use Hoare logic to verify the *partial correctness* of the following program with respect to the specification: [8 marks]

```
Precondition: X >= 0 ∧ Y >= 0
Postcondition: z == X*Y

Program:    int i = 0;
            int z = 0;
            while (i != Y)
              i = i + 1;
              z = z + X;
```

Clearly identify the Hoare logic rules and simplification steps used at each stage of the verification.

(c) Describe the difference between verifying the *partial* and *total correctness* of a loop. Clearly identify the extra conditions that must be specified and verified. Show how you would verify the total correctness of the program in (b) above. [4 marks]

(d) Explain what is meant by the term *under-specification* in program verification. Support your solution by providing a correct specification for a program which sorts an array. Explain why your solution is not under-specified. [5 marks]

[25 marks]

2 (a) With reference to the Dafny code provided below: [8 marks]

i. Provide preconditions and variant functions for `Sum0` and `Sum1` ghost functions.
ii. Provide a precondition, a variant function and a postcondition for the `Sum2` lemma below.
iii. Explain the role of ghost code in program verification.
iv. Explain the role of lemmas in program verification.

```
function F(x: int): int

ghost function Sum0(lo: int, hi: int): int{
  if lo == hi then 0 else F(lo) + Sum0(lo + 1, hi)
}

ghost function Sum1(lo: int, hi: int): int{
  if lo == hi then 0 else
                Sum1(lo, hi - 1) + F(hi - 1)
}

lemma Sum2(lo: int, hi: int){
  if lo != hi {
    PrependSumDown(lo, hi);
  }
}
lemma PrependSumDown(lo: int, hi: int)
  requires lo < hi
  ensures F(lo) + Sum0(lo + 1, hi) == Sum1(lo, hi)
  decreases hi - lo
{
}
```

2 (b) The following Dafny method returns the minimum value in an array, or 0 if the array is empty. [8 marks]

i. Provide a variant and an invariant for the loop verification. Explain how they are used in the verification.
ii. Describe and explain the expected verification error if the initialisation `min := a[0];` is changed to `min := 0;`
iii. Describe and explain the expected verification error if the initialisation `min := 0;` is changed to `min := a[0];`

```
method min_elt(a:array<int>) returns(min:int)
ensures
    if (a.Length == 0) then min == 0
    else
      (forall i :: 0<=i<a.Length ==> a[i]>=min) &&
       (exists i :: 0 <=i<a.Length && min == a[i])
{
   if (a.Length == 0)  { min := 0;}
   else {
            min := a[0];
            var i:int := 1;
            while (i < a.Length)
            {
               if(a[i] < min) {
                   min := a[i];
               }
               i := i + 1;
            }
        }
}
```

(c) Specify and implement a `class Queue` with the normal LIFO queue behaviour with operations to: [9 marks]

i. create a queue object.
ii. add an element to the queue.
iii. remove an element from the queue.

Use your code to illustrate how verifiers (such as Dafny, OpenJML and Why3) check class invariants, constructor contracts and method contracts.

[25 marks]

3 (a) Explain how system modeling and analysis is supported by the *Event-B modelling system*. Your answer should include an explanation of *contexts, machines, invariants, events* and *before-after predicates*. Support your answer with an appropriate example listing at least two proof obligations that must be proved for the model to be correct. [8 marks]

(b) Define what is meant by the term *Data Refinement*. Support your answer with an example. Explain how both *Dafny* and *Event-B* support data refinement. [6 marks]

(c) How does the SMT solver *Z3* determine if a model exists that satisfies the following set of equations? Explain what property is being checked below and what each line of code achieves. [5 marks]

```
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)
(assert (> a 4))
(assert (< b 0))
(assert (> c 2))
(assert (= (+ (* a a) (* b b)) (* c c)))
(check-sat)
(get-model)
```

(d) SAT solvers can be classified into those based on *Conflict-Driven Clause Learning (CDCL)* and those based on a *stochastic search*. Explain using an example how the CDCL algorithm operates. What advantage does CDCL have over stochastic search? [6 marks]

[25 marks]

4 (a) Explain how the *Spin model checker* verifies temporal properties of a system. Your answer should explain the representation of the system model, the representation of the formula to be checked, expressing invariant properties and using labels to verify properties of the model. [8 marks]

Provide examples to support your answer.

(b) Specify the following properties in *Linear Temporal Logic (LTL)*. Explain your answers. [10 marks]

i. if the coffee level is low in state s_i, $i <= 0$ then at state s_{i+1} an alarm will ring.
ii. receiving a good salary and enjoying your job is possible.
iii. if the door is locked at state s_i, then at some future state s_j, $j >= i$ the door will unlock.
iv. if the shop is open and the front of a queue is served at time s_i, then the shop will start serving the queue at some future time s_j, $j >= i$, and continue serving until the end of the queue is reached at some state s_k, $k > j$.

(c) Explain the difference between *Linear Temporal Logic (LTL)* and *Computation Tree Logic (CTL).* Support your explanation by discussing their use when expressing: [7 marks]

i. *safety properties.*
ii. *liveness properties.*
iii. *fairness properties.*